Introdução a

Criptografia de Chave-Pública

Prof. Dr. Iaçanã Ianiski Weber

Confiabilidade e Segurança de Software

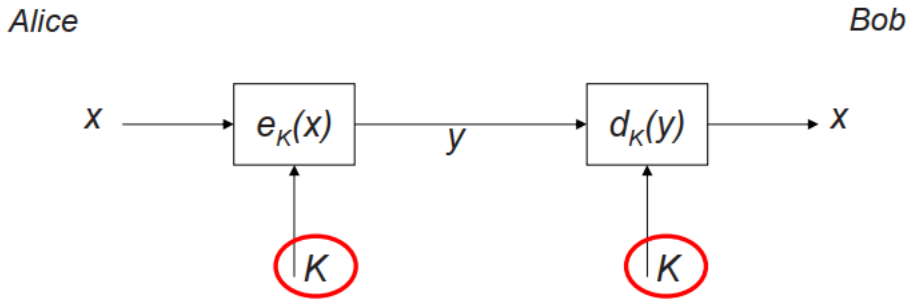
98G08-4

Agradecimentos especiais ao Prof. Avelino Zorzo e aos Autores Christof Paar e Jan Pelzl pelo material.

Índice

Índice

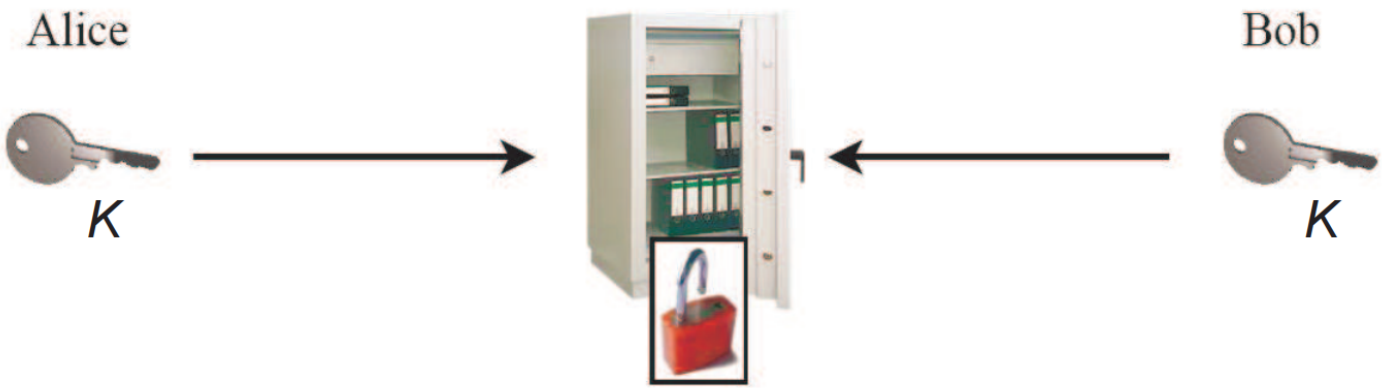
Criptografia Simétrica revisitada



Duas propriedades dos sistemas criptográficos simétricos (de chave secreta):

- A **mesma chave secreta** K é usada para a cifragem e a decifragem
- As funções de cifragem e decifragem são muito semelhantes (ou até mesmo idênticas)

Criptografia Simétrica: Analogia



Cofre com uma fechadura forte: apenas Alice e Bob possuem uma cópia da chave

- Alice cifra $\Rightarrow$ tranca a mensagem no cofre usando sua chave
- Bob decifra $\Rightarrow$ usa sua cópia da chave para abrir o cofre

Criptografia Simétrica: Limitações

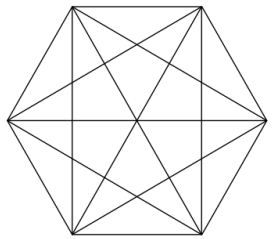
- Algoritmos simétricos, como o AES, são muito seguros, rápidos e amplamente utilizados, **mas**:
  - Problema de distribuição de chaves: a chave secreta deve ser **transportada com segurança**
  - Número de chaves: em uma rede, cada par de usuários precisa de uma chave individual

$\Rightarrow$ Uma rede com n usuários requer $\frac{n \cdot (n-1)}{2}$ chaves, e cada usuário armazena $(n-1)$ chaves

Exemplo:

6 usuários (nós)

$\frac{6 \cdot 5}{2} = 15$ chaves (arestas)



- Alice e Bob podem tentar se fazer passar um pelo outro, pois possuem chaves idênticas.
  Para evitar isso precisamos da propriedade de **"não-repúdio"**

Ideia por trás da Criptografia Assimétrica

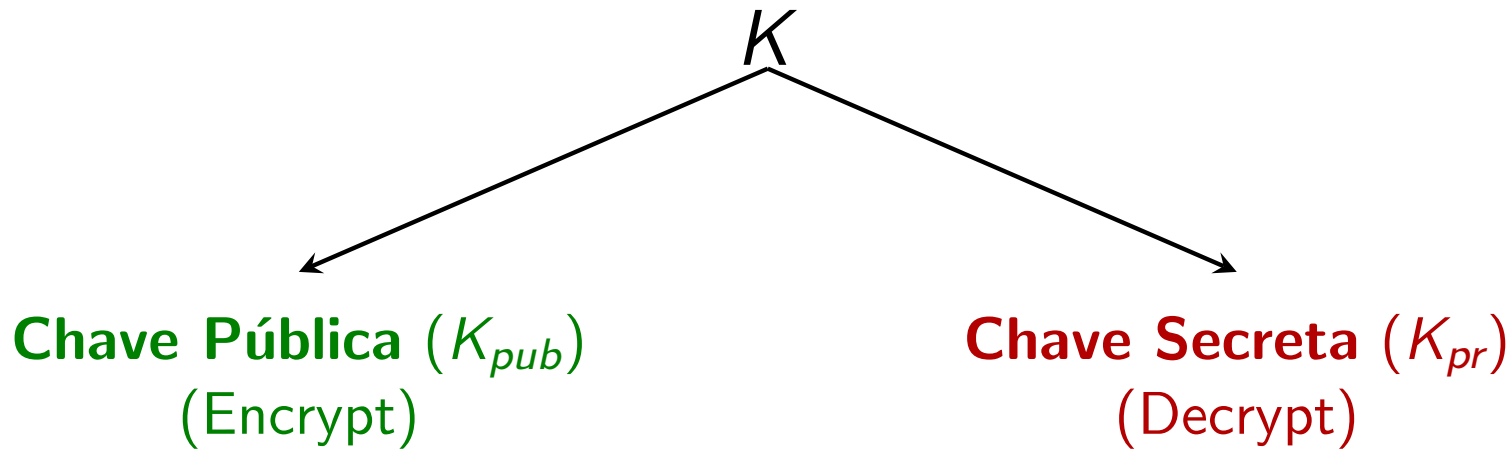
Nova ideia:
Usar o princípio da *"caixa de correio"*:

- **Qualquer pessoa** pode inserir uma carta
- Mas **apenas o dono** possui a chave correta para abrir a caixa

1976: primeira publicação de um algoritmo desse tipo por Whitfield Diffie e Martin Hellman. Também houveram contribuições por Ralph Merkle.

Criptografia Assimétrica (Chave Pública)

Princípio: "Dividir" a chave



$\Rightarrow$ Durante a geração de chaves, um par de chaves K_{pub} e K_{pr} é computado

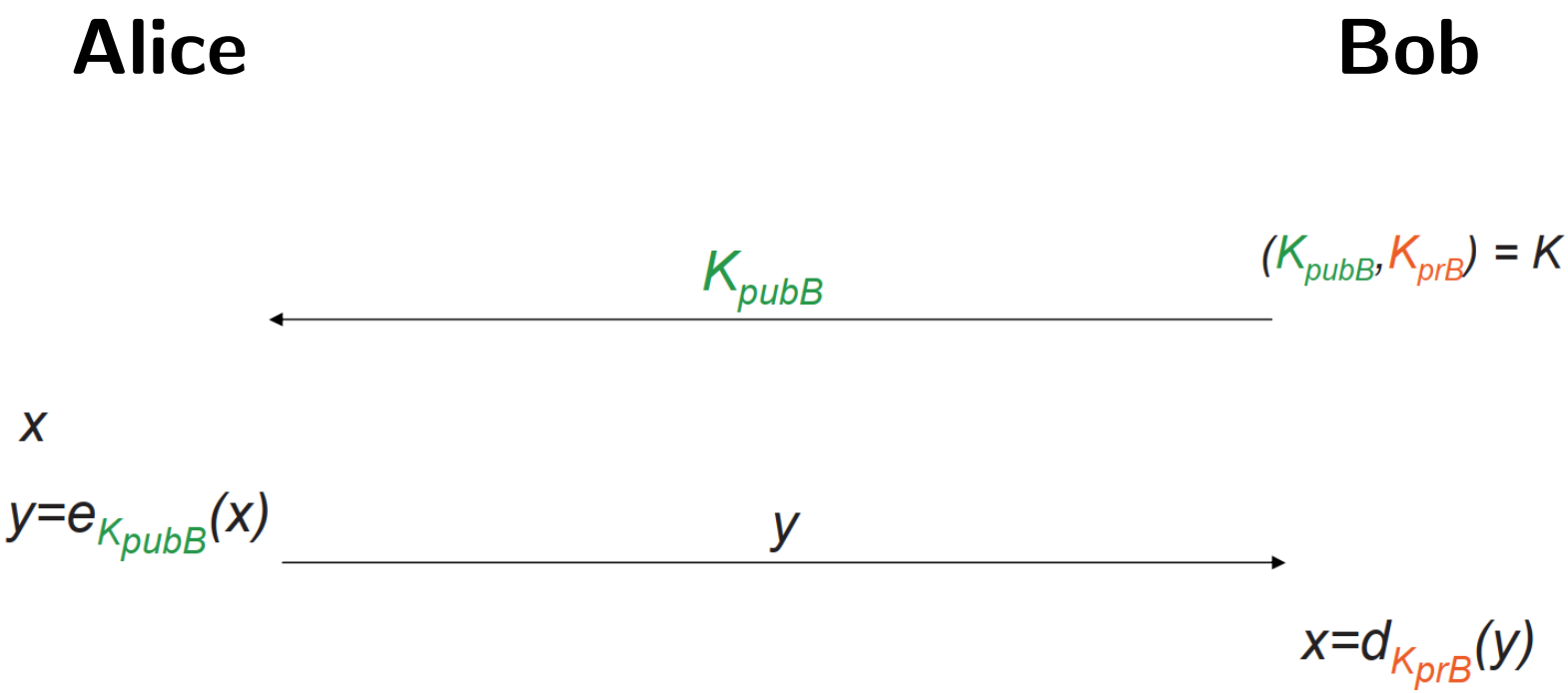
Criptografia Assimétrica: Analogia

Cofre com fechadura pública e fechadura privada:

- Alice deposita (cifra) uma mensagem utilizando a *chave pública* K_{pub} — que **não é secreta**
- Somente Bob possui a *chave privada* K_{pr} — que **é secreta** — para recuperar (decifrar) a mensagem

Índice

Protocolo Básico de Criptografia com Chave Pública



$\Rightarrow$ Problema de Distribuição de Chaves resolvido

Mecanismos de Segurança da Criptografia de Chave Pública

Aqui estão os principais mecanismos que podem ser realizados com criptografia assimétrica:

- **Distribuição de Chaves** (ex.: troca de chaves Diffie-Hellman, RSA) sem a necessidade de um segredo (chave) pré-compartilhado
- **Não-repúdio e Assinaturas Digitais** (ex.: RSA, DSA ou ECDSA) para garantir a integridade da mensagem
- **Identificação**, usando protocolos de desafio-resposta com assinaturas digitais
- **Cifragem** (ex.: RSA)
  Desvantagem: Computacionalmente muito intensiva
  (*1000 vezes mais lenta que algoritmos simétricos!*)

Protocolo Básico de Transporte de Chaves 1/2

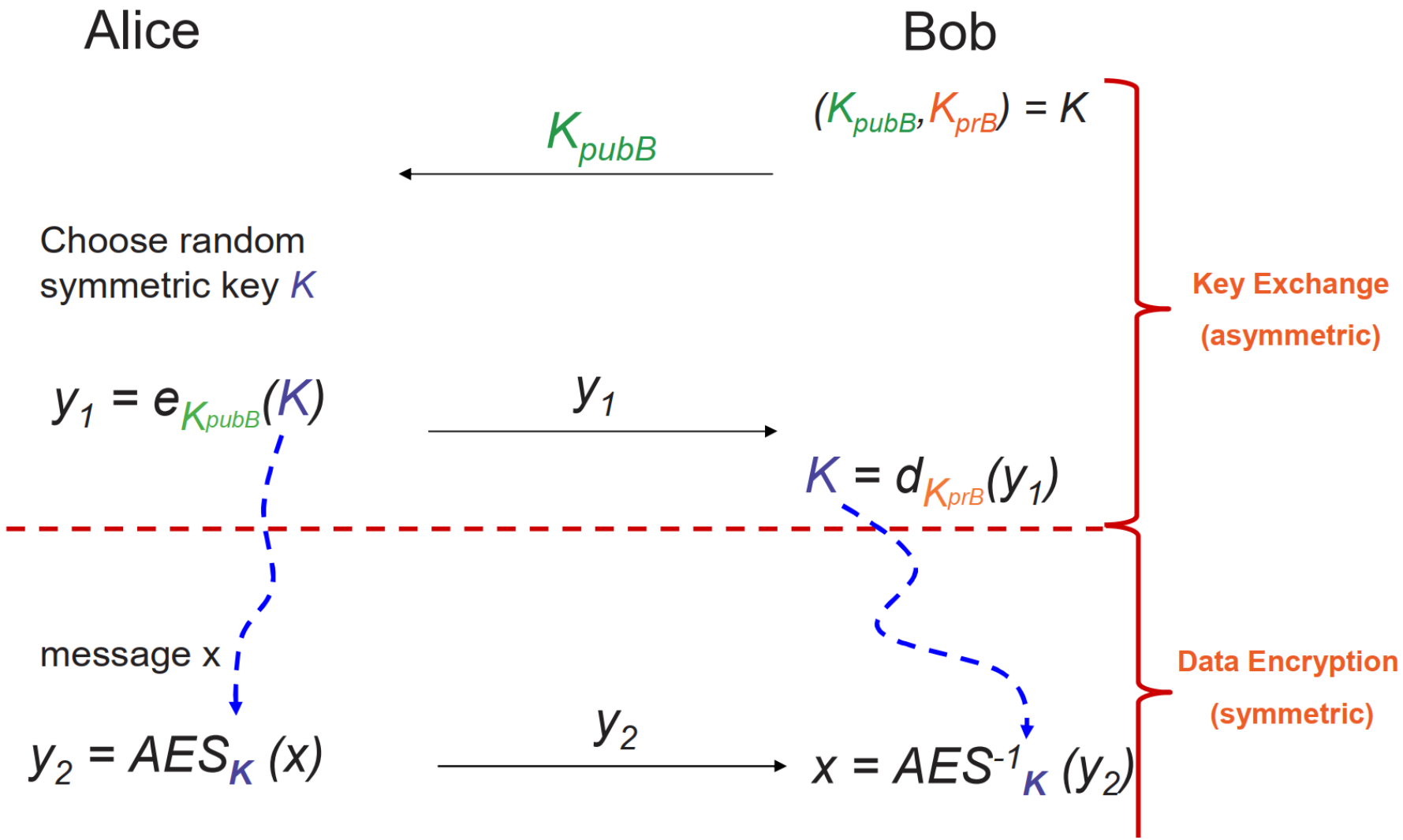
Na prática usamos **sistemas híbridos**, incorporando tanto algoritmos assimétricos quanto simétricos

1. **Troca de chaves** (para esquemas simétricos) e **assinaturas digitais** são realizadas com algoritmos **assimétricos** (lentos)

2. **Cifragem** dos dados é feita com cifras **simétricas** (rápidas), por exemplo, *block ciphers* ou *stream ciphers*

Protocolo Básico de Transporte de Key 2/2

Exemplo: Protocolo híbrido com AES como cifra simétrica



Índice

Como construir Algoritmos de Chave Pública

Esquemas assimétricos são baseados em uma "one-way function" (função de mão única) $f()$:

- Calcular $y = f(x)$ é computacionalmente fácil
- Calcular $x = f^{-1}(y)$ é computacionalmente inviável

One-way functions são baseadas em problemas matematicamente difíceis.

Três famílias principais:

- **Fatoração de inteiros** (RSA):

  Dado um inteiro composto n, encontrar seus fatores primos (Multiplicar dois primos: fácil)

- **Logaritmo Discreto** (Diffie-Hellman, DSA):

  Dados a, y e m, encontrar x tal que $a^x \equiv y \pmod{m}$ (Exponenciação a^x: fácil)

- **Curvas Elípticas (EC)** (ECDH, ECDSA): Generalização do logaritmo discreto

Nota: Os problemas são considerados matematicamente difíceis, mas não existe prova (até o momento).

Criptografia Pós-Quântica (PQC): Problemas Difíceis

A Criptografia Pós-Quântica (PQC) busca algoritmos resistentes a computadores quânticos, baseando-se em outros problemas matemáticos considerados difíceis: **Principais famílias de problemas para PQC:**

- **Baseada em Reticulados (Lattice-based)**: Encontrar vetores específicos (e.g., o mais curto) em um reticulado.
  - FIPS 203 - CRYSTALS - Kyber - **ML-KEM**
  - FIPS 204 - CRYSTALS - Dilithium - **ML-DSA**
  - FIPS 206 - FALCON - **FN-DSA**
- **Baseada em Códigos (Code-based)**: Decodificar um código linear geral na presença de erros.
  - FIPS ? - Hamming Quasi-Cyclic (HQC) - **?-KEM**
- **Baseada em Hashes (Hash-based Signatures)**: Inverter uma função hash (achar pré-imagem) ou encontrar colisões.
  - FIPS 205 - SPHINCS+ - **SLH-DSA**

Nota: Assim como na criptografia clássica, a intratabilidade destes problemas é conjecturada, mas não universalmente provada.

Tamanhos de Chave e Níveis de Segurança

Simétrico	*ECC*	*RSA*	*Observação*
128 Bit	256 Bit	3072 Bit	Segurança moderna padrão, adequada para proteção de dados por vários anos.
192 Bit	384 Bit	7680 Bit	Segurança elevada para dados sensíveis ou proteção de longo prazo.
256 Bit	512 Bit	15360 Bit	Segurança de muito longo prazo (computadores clássicos), para dados altamente sensíveis.

- A complexidade exata do RSA (fatoração) é difícil de estimar.
- A existência de computadores quânticos provavelmente seria o fim para ECC, RSA & DL (pelo menos daqui a 2-3 décadas, e algumas pessoas duvidam que CQs venham a existir).

Índice

Algoritmo de Euclides 1/2

- Calcule o máximo divisor comum $\text{mdc}(r_0, r_1)$ de dois inteiros r_0 e r_1.
- mdc é fácil para números pequenos:
  1. fatore r_0 e r_1
  2. mdc = maior fator comum
- Exemplo:

$$r_0 = 84 = 2^2 \cdot 3 \cdot 7$$
$$r_1 = 30 = 2 \cdot 3 \cdot 5$$

$\rightarrow$ O mdc é o produto de todos os fatores primos comuns:

$$2 \cdot 3 = 6 = \text{mdc}(30, 84)$$

- Mas **fatorar é complicado** (e frequentemente inviável) para números grandes.

Algoritmo de Euclides 2/2

- Observação: $\mathbf{mdc}(r_0, r_1) = \mathbf{mdc}(r_0 - r_1, r_1)$

$\rightarrow$ Ideia central:

- Reduzir o problema de encontrar o mdc de dois números dados ao do **mdc de dois números menores**
- Repetir o processo recursivamente
- O $\mathbf{mdc}(r_i, \mathbf{0}) = r_i$ final é a resposta para o problema original !

Exemplo: $\text{mdc}(r_0, r_1)$ para $r_0 = 27$ e $r_1 = 21$

$$\text{mdc}(27, 21) = \text{mdc}(1 \cdot 21 + 6, 21) = \text{mdc}(21, 6)$$
$$\text{mdc}(21, 6) = \text{mdc}(3 \cdot 6 + 3, 6) = \text{mdc}(6, 3)$$
$$\text{mdc}(6, 3) = \text{mdc}(2 \cdot 3 + 0, 3) = \text{mdc}(3, 0) = 3$$

- Nota: esse método é muito eficiente mesmo para números longos, pois a complexidade cresce **linearmente** com o número de bits.

Algoritmo Estendido de Euclides 1/2

- Estende o algoritmo de Euclides para encontrar o **inverso modular** de $r_1 \pmod{r_0}$.
- O EEA (Algoritmo de Euclides Estendido) calcula s, t, e o mdc:

$$\text{mdc}(r_0, r_1) = s \cdot r_0 + t \cdot r_1$$

- Tome a relação mod r_0. Se $\text{mdc}(r_0, r_1) = 1$, então $s \cdot r_0 + t \cdot r_1 = 1$:

$$\begin{aligned} s \cdot r_0 + t \cdot r_1 &= 1 \\ s \cdot 0 + t \cdot r_1 &\equiv 1 \pmod{r_0} \\ r_1 \cdot t &\equiv 1 \pmod{r_0} \end{aligned}$$

  Desta forma, temos que: **t é o inverso de** $r_1 \pmod{r_0}$
- Note que $\text{mdc}(r_0, r_1) = 1$ para que o inverso exista.
- **Fórmulas recursivas** para calcular s e t em cada passo.

Função Phi de Euler 1/2

- Novo problema, importante para sistemas de chave pública, ex.: RSA:
  Dado o conjunto dos m inteiros $\{0, 1, 2, \ldots, m-1\}$,
  Quantos números no conjunto são **primos relativos a** m?
- Resposta: Função Phi de Euler $\Phi(m)$.
- **Exemplo** para os conjuntos $\{0, 1, 2, 3, 4, 5\}$ $(m = 6)$, e $\{0, 1, 2, 3, 4\}$ $(m = 5)$:

$\underline{m = 6}$:

$\text{mdc}(0, 6) = 6$

$\text{mdc}(1, 6) = 1 \quad \leftarrow$

$\text{mdc}(2, 6) = 2$

$\text{mdc}(3, 6) = 3$

$\text{mdc}(4, 6) = 2$

$\text{mdc}(5, 6) = 1 \quad \leftarrow$

$\boldsymbol{\Phi(6) = 2}$.

$\underline{m = 5}$:

$\text{mdc}(0, 5) = 5$

$\text{mdc}(1, 5) = 1 \quad \leftarrow$

$\text{mdc}(2, 5) = 1 \quad \leftarrow$

$\text{mdc}(3, 5) = 1 \quad \leftarrow$

$\text{mdc}(4, 5) = 1 \quad \leftarrow$

$\boldsymbol{\Phi(5) = 4}$.

Função Phi de Euler 2/2

- Calcular Phi é especialmente fácil para $m = p \cdot q$ (onde p, q são primos distintos):

$$\Phi(m) = (p-1)(q-1)$$

- Exemplo para $m = 899 = 29 \cdot 31$:

$$\Phi(899) = (29-1) \cdot (31-1) = 28 \cdot 30 = \mathbf{840}$$

- Nota: Encontrar $\Phi(m)$ é computacionalmente fácil **se a fatoração de m é conhecida** (caso contrário, o cálculo de $\Phi(m)$ torna-se computacionalmente inviável para números grandes).

Lições Aprendidas

- Algoritmos de chave pública possuem **capacidades que cifras simétricas não têm**, em particular assinatura digital e funções de estabelecimento de chaves (key establishment).
- Algoritmos de chave pública são **computacionalmente intensivos** (uma forma elegante de dizer que são *lentos*), e portanto são pouco adequados para criptografia de grandes volumes de dados (bulk data encryption).
- Apenas **três famílias de esquemas de chave pública** são amplamente utilizadas. Isto é consideravelmente menos do que no caso de algoritmos simétricos.
- O **algoritmo Euclidiano estendido** nos permite calcular **inversos modulares** rapidamente, o que é importante para quase todos os esquemas de chave pública.
- A **função phi de Euler** (Φ) nos dá o número de elementos menores que um inteiro n que são primos relativos a n. Isto é importante para o esquema de criptografia RSA.